Environmental Bisimulations for Delimited-Control Operators

Dariusz Biernacki, Sergueï Lenglet

Environmental Bisimulations for Delimited-Control Operators

Dariusz Biernacki[1] and Sergueï Lenglet[2]

[1] Institute of Computer Science, University of Wrocław
[2] LORIA, Université de Lorraine

Abstract. We present a theory of environmental bisimilarity for the delimited-control operators *shift* and *reset*. We consider two different notions of contextual equivalence: one that does not require the presence of a top-level control delimiter when executing tested terms, and another one, fully compatible with the original CPS semantics of shift and reset, that does. For each of them, we develop sound and complete environmental bisimilarities, and we discuss up-to techniques.

1 Introduction

Control operators for delimited continuations [8, 10] provide elegant means for expressing advanced control mechanisms [8, 12]. Moreover, they play a fundamental role in the semantics of computational effects [11], normalization by evaluation [2] and as a crucial refinement of abortive control operators such as *callcc* [10, 21]. Of special interest are the control operators *shift* and *reset* [8] due to their origins in continuation-passing style (CPS) and their connection with computational monads – as demonstrated by Filinski [11], shift and reset can express in direct style arbitrary computational effects, such as mutable state, exceptions, etc. Operationally, the control delimiter reset delimits the current continuation and the control operator shift abstracts the current delimited continuation as a first class value that when resumed is composed with the then-current continuation.

Because of the complex nature of control effects, it can be difficult to determine if two programs that use shift and reset are equivalent (i.e., behave in the same way) or not. *Contextual equivalence* [17] is widely considered as the most natural equivalence on terms in languages similar to the λ-calculus. Roughly, two terms are contextually equivalent if we cannot tell them apart when they are executed within any context. The latter quantification over contexts makes this relation hard to use in practice, so we usually look for simpler characterizations of contextual equivalence, such as coinductively defined *bisimilarities*.

In our previous work, we defined *applicative* [4] and *normal form* [5] bisimilarities for shift and reset. Applicative bisimilarity characterizes contextual equivalence, but still quantifies over some contexts to relate terms (e.g., λ-abstractions are applied to the same arbitrary argument). As a result, some equivalences remain quite difficult to prove. In contrast, normal form bisimilarity does not

contain any quantification over contexts or arguments in its definition: the tested terms are reduced to normal forms, which are then decomposed in bisimilar subterms. Consequently, proofs of equivalence are usually simpler than with applicative bisimilarity, and they can be simplified even further with *up-to techniques*. However, normal form bisimilarity is not *complete*, i.e., there exist contextually equivalent terms which are not normal form bisimilar.

Environmental bisimilarity [19] is a different kind of behavioral equivalence which in terms of strength and practicality can be situated in between applicative and normal form bisimilarities. It has originally been proposed in [23] and has been since defined in various higher-order languages (see, e.g., [20, 22, 18]). Like applicative bisimilarity, it uses some particular contexts to test terms, except that the testing contexts are built from an environment, which represents the knowledge built so far by an outside observer. Environmental bisimilarity usually characterizes contextual equivalence, but is harder to establish than applicative bisimilarity. Nonetheless, like with normal form bisimilarity, one can define powerful up-to techniques [19] to simplify the equivalence proofs. Besides, the authors of [15] argue that the additional complexity of environmental bisimilarity is necessary to handle more realistic features, like local state or exceptions.

In the quest for a powerful enough (i.e., as discriminative as contextual equivalence) yet easy-to-use equivalence for delimited control, we study in this paper the environmental theory of a calculus with shift and reset. More precisely, we consider two semantics for shift and reset: the original one [3], where terms are executed within a top-level reset, and a more relaxed semantics where this requirement is lifted. The latter is commonly used in implementations of shift and reset [9, 11] as well as in some studies of these operators [1, 13], including our previous work [4, 5]. So far, the behavioral theory of shift and reset with the original semantics has not been studied. Firstly, we define environmental bisimilarity for the relaxed semantics and study its properties; especially we discuss the problems raised by delimited control for the definition of bisimulation up to context, one of the most powerful up-to techniques. Secondly, we propose the first behavioral theory for the original semantics, and we pinpoint the differences between the equivalences of the two semantics. In particular, we show that the environmental bisimilarity for the original semantics is complete w.r.t. the axiomatization of shift and reset of [14], which is not the case for the relaxed semantics, as already proved in [4] for applicative bisimilarity.

In summary, we make the following contributions in this paper.

- We show that environmental bisimilarity can be defined for a calculus with delimited control, for which we consider two different semantics. In each case, the defined bisimilarity equals contextual equivalence.
- For the relaxed semantics, we explain how to handle *stuck terms*, i.e., terms where a capture cannot go through because of the lack of an outermost reset.
- We discuss the limits of the usual up-to techniques in the case of delimited control.
- For the original semantics, we define a contextual equivalence, and a corresponding environmental bisimilarity. Proving soundness of the bisimilar-

ity w.r.t. contextual equivalence requires significant changes from the usual soundness proof scheme. We discuss how environmental bisimilarity is easier to adapt than applicative bisimilarity.
– We give examples illustrating the differences between the two semantics.

The rest of the paper is organized as follows: in Section 2, we present the calculus $\lambda_{\mathcal{S}}$ used in this paper, and recall some results, including the axiomatization of [14]. We develop an environmental theory for the relaxed semantics in Section 3, and for the original semantics in Section 4. We conclude in Section 5. An extended version of this article [6] contains most of the omitted proofs.

2 The Calculus $\lambda_{\mathcal{S}}$

2.1 Syntax

The language $\lambda_{\mathcal{S}}$ extends the call-by-value λ-calculus with the delimited-control operators *shift* and *reset* [8]. We assume we have a set of term variables, ranged over by x, y, z, and k. We use k for term variables representing a continuation (e.g., when bound with a shift), while x, y, and z stand for any values; we believe such distinction helps to understand examples and reduction rules. The syntax of terms is given by the following grammar:

$$\text{Terms: } t ::= x \mid \lambda x.t \mid t\ t \mid \mathcal{S}k.t \mid \langle t \rangle$$

Values, ranged over by v, are terms of the form $\lambda x.t$. The operator shift ($\mathcal{S}k.t$) is a capture operator, the extent of which is determined by the delimiter reset ($\langle\cdot\rangle$). A λ-abstraction $\lambda x.t$ binds x in t and a shift construct $\mathcal{S}k.t$ binds k in t; terms are equated up to α-conversion of their bound variables. The set of free variables of t is written $\mathsf{fv}(t)$; a term t is *closed* if $\mathsf{fv}(t) = \emptyset$.

We distinguish several kinds of contexts, represented outside-in, as follows:

$$\begin{array}{ll} \text{Pure contexts:} & E ::= \Box \mid v\ E \mid E\ t \\ \text{Evaluation contexts:} & F ::= \Box \mid v\ F \mid F\ t \mid \langle F \rangle \\ \text{Contexts:} & C ::= \Box \mid \lambda x.C \mid t\ C \mid C\ t \mid \mathcal{S}k.C \mid \langle C \rangle \end{array}$$

Regular contexts are ranged over by C. The pure evaluation contexts[3] (abbreviated as pure contexts), ranged over by E, represent delimited continuations and can be captured by shift. The call-by-value evaluation contexts, ranged over by F, represent arbitrary continuations and encode the chosen reduction strategy. Filling a context C (respectively E, F) with a term t produces a term, written $C[t]$ (respectively $E[t]$, $F[t]$); the free variables of t may be captured in the process. We extend the notion of free variables to contexts (with $\mathsf{fv}(\Box) = \emptyset$), and we say a context C (respectively E, F) is *closed* if $\mathsf{fv}(C) = \emptyset$ (respectively $\mathsf{fv}(E) = \emptyset$, $\mathsf{fv}(F) = \emptyset$).

[3] This terminology comes from Kameyama (e.g., in [14]).

2.2 Reduction Semantics

The call-by-value reduction semantics of $\lambda_{\mathcal{S}}$ is defined as follows, where $t\{v/x\}$ is the usual capture-avoiding substitution of v for x in t:

$$
\begin{array}{ll}
(\beta_v) & F[(\lambda x.t)\ v] \to_{\mathrm{v}} F[t\{v/x\}] \\
(\mathit{shift}) & F[\langle E[\mathcal{S}k.t]\rangle] \to_{\mathrm{v}} F[\langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle] \text{ with } x \notin \mathsf{fv}(E) \\
(\mathit{reset}) & F[\langle v\rangle] \to_{\mathrm{v}} F[v]
\end{array}
$$

The term $(\lambda x.t)\ v$ is the usual call-by-value redex for β-reduction (rule (β_v)). The operator $\mathcal{S}k.t$ captures its surrounding context E up to the dynamically nearest enclosing reset, and substitutes $\lambda x.\langle E[x]\rangle$ for k in t (rule (*shift*)). If a reset is enclosing a value, then it has no purpose as a delimiter for a potential capture, and it can be safely removed (rule (*reset*)). All these reductions may occur within a metalevel context F, so the reduction rules specify both the notion of reduction and the chosen call-by-value evaluation strategy that is encoded in the grammar of the evaluation contexts. Furthermore, the reduction relation $\to_{\mathrm{v}}$ is compatible with evaluation contexts F, i.e., $F[t] \to_{\mathrm{v}} F[t']$ whenever $t \to_{\mathrm{v}} t'$.

There exist terms which are not values and which cannot be reduced any further; these are called *stuck terms*.

Definition 1. *A term t is stuck if t is not a value and $t \not\to_{\mathrm{v}}$.*

For example, the term $E[\mathcal{S}k.t]$ is stuck because there is no enclosing reset; the capture of E by the shift operator cannot be triggered.

Lemma 1. *A closed term t is stuck iff $t = E[\mathcal{S}k.t']$ for some E, k, and t'.*

Definition 2. *A term t is a normal form if t is a value or a stuck term.*

We call *redexes* (ranged over by r) terms of the form $(\lambda x.t)\ v$, $\langle E[\mathcal{S}k.t]\rangle$, and $\langle v\rangle$. Thanks to the following unique-decomposition property, the reduction relation $\to_{\mathrm{v}}$ is deterministic.

Lemma 2. *For all closed terms t, either t is a normal form, or there exist a unique redex r and a unique context F such that $t = F[r]$.*

Finally, we write $\to_{\mathrm{v}}^*$ for the transitive and reflexive closure of $\to_{\mathrm{v}}$, and we define the evaluation relation of $\lambda_{\mathcal{S}}$ as follows.

Definition 3. *We write $t \Downarrow_{\mathrm{v}} t'$ if $t \to_{\mathrm{v}}^* t'$ and $t' \not\to_{\mathrm{v}}$.*

The result of the evaluation of a closed term, if it exists, is a normal form. If a term t admits an infinite reduction sequence, we say it *diverges*, written $t \Uparrow_{\mathrm{v}}$. Henceforth, we use $\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x)$ as an example of such a term.

2.3 CPS Equivalence

In [14], the authors propose an equational theory of shift and reset based on CPS [8]. The idea is to relate terms that have $\beta\eta$-convertible CPS translations.

Definition 4. *Terms t_0 and t_1 are CPS equivalent, written $t_0 \equiv t_1$, if their CPS translations are $\beta\eta$-convertible.*

Kameyama and Hasegawa propose eight axioms in [14] to characterize CPS equivalence: two terms are CPS equivalent iff one can derive their equality using the equations below. Note that the axioms are defined on open terms, and suppose variables as values.

$$(\lambda x.t)\, v =_{\text{KH}} t\{v/x\} \qquad (\lambda x.E[x])\, t =_{\text{KH}} E[t] \text{ if } x \notin \mathsf{fv}(E)$$

$$\langle E[\mathcal{S}k.t]\rangle =_{\text{KH}} \langle t\{\lambda x.\langle E[x]\rangle/k\}\rangle \qquad \langle(\lambda x.t_0)\,\langle t_1\rangle\rangle =_{\text{KH}} (\lambda x.\langle t_0\rangle)\,\langle t_1\rangle$$

$$\langle v\rangle =_{\text{KH}} v \qquad \mathcal{S}k.\langle t\rangle =_{\text{KH}} \mathcal{S}k.t$$

$$\lambda x.v\, x =_{\text{KH}} v \text{ if } x \notin \mathsf{fv}(v) \qquad \mathcal{S}k.k\, t =_{\text{KH}} t \text{ if } k \notin \mathsf{fv}(t)$$

We use the above relations as examples throughout the paper. Of particular interest is the axiom $(\lambda x.E[x])\, t =_{\text{KH}} E[t]$ (if $x \notin \mathsf{fv}(E)$), called β_Ω in [14], which can be difficult to prove with bisimilarities [4].

2.4 Context Closures

Given a relation $\mathcal{R}$ on terms, we define two context closures that generate respectively terms and evaluation contexts. The term generating closure $\widehat{\mathcal{R}}$ is defined inductively as the smallest relation satisfying the following rules:

$$\frac{t\ \mathcal{R}\ t'}{t\ \widehat{\mathcal{R}}\ t'} \qquad x\ \widehat{\mathcal{R}}\ x \qquad \frac{t\ \widehat{\mathcal{R}}\ t'}{\lambda x.t\ \widehat{\mathcal{R}}\ \lambda x.t'} \qquad \frac{t_0\ \widehat{\mathcal{R}}\ t_0' \quad t_1\ \widehat{\mathcal{R}}\ t_1'}{t_0\, t_1\ \widehat{\mathcal{R}}\ t_0'\, t_1'} \qquad \frac{t\ \widehat{\mathcal{R}}\ t'}{\mathcal{S}k.t\ \widehat{\mathcal{R}}\ \mathcal{S}k.t'} \qquad \frac{t\ \widehat{\mathcal{R}}\ t'}{\langle t\rangle\ \widehat{\mathcal{R}}\ \langle t'\rangle}$$

Even if $\mathcal{R}$ is defined only on closed terms, $\widehat{\mathcal{R}}$ is defined on open terms. In this paper, we consider the restriction of $\widehat{\mathcal{R}}$ to closed terms unless stated otherwise. The context generating closure $\widetilde{\mathcal{R}}$ of a relation $\mathcal{R}$ is defined inductively as the smallest relation satisfying the following rules:

$$\frac{}{\Box\ \widetilde{\mathcal{R}}\ \Box} \qquad \frac{F_0\ \widetilde{\mathcal{R}}\ F_1 \qquad v_0\ \widehat{\mathcal{R}}\ v_1}{v_0\ F_0\ \widetilde{\mathcal{R}}\ v_1\ F_1} \qquad \frac{F_0\ \widetilde{\mathcal{R}}\ F_1 \qquad t_0\ \widehat{\mathcal{R}}\ t_1}{F_0\ t_0\ \widetilde{\mathcal{R}}\ F_1\ t_1} \qquad \frac{F_0\ \widetilde{\mathcal{R}}\ F_1}{\langle F_0\rangle\ \widetilde{\mathcal{R}}\ \langle F_1\rangle}$$

Again, we consider only the restriction of $\widetilde{\mathcal{R}}$ to closed contexts.

3 Environmental Relations for the Relaxed Semantics

In this section, we define an environmental bisimilarity which characterizes the contextual equivalence of [4, 5], where stuck terms can be observed.

3.1 Contextual Equivalence

We recall the definition of contextual equivalence $\approx_c$ for the relaxed semantics (given in [4]).

Definition 5. *For all t_0, t_1 be terms. We write $t_0 \approx_c t_1$ if for all C such that $C[t_0]$ and $C[t_1]$ are closed, the following hold:*

- *$C[t_0] \Downarrow_{\mathrm{v}} v_0$ implies $C[t_1] \Downarrow_{\mathrm{v}} v_1$;*
- *$C[t_0] \Downarrow_{\mathrm{v}} t'_0$, where t'_0 is stuck, implies $C[t_1] \Downarrow_{\mathrm{v}} t'_1$, with t'_1 stuck as well;*

and conversely for $C[t_1]$.

The definition is simpler when using the following context lemma [16] (for a proof see Section 3.4 in [4]). Instead of testing with general, closing contexts, we can close the terms with values and then put them in evaluation contexts.

Lemma 3 (Context Lemma). *We have $t_0 \approx_c t_1$ iff for all closed contexts F and for all substitutions σ (mapping variables to closed values) such that $t_0\sigma$ and $t_1\sigma$ are closed, the following hold:*

- *$F[t_0\sigma] \Downarrow_{\mathrm{v}} v_0$ implies $F[t_1\sigma] \Downarrow_{\mathrm{v}} v_1$;*
- *$F[t_0\sigma] \Downarrow_{\mathrm{v}} t'_0$, where t'_0 is stuck, implies $F[t_1\sigma] \Downarrow_{\mathrm{v}} t'_1$, with t'_1 stuck as well;*

and conversely for $F[t_1\sigma]$.

In [4], we prove that $\approx_c$ satisfies all the axioms of CPS equivalence except for $\mathcal{S}k.k\, t =_{\mathrm{KH}} t$ (provided $k \notin \mathsf{fv}(t)$): indeed, $\mathcal{S}k.k\, t$ is stuck, but t may evaluate to a value. Conversely, some contextually equivalent terms are not CPS equivalent, like Turing's and Church's call-by-value fixed point combinators. Similarly, two arbitrary diverging terms are related by $\approx_c$, but not necessarily by $\equiv$.

3.2 Definition of Environmental Bisimulation and Basic Properties

Environmental bisimulations use an environment $\mathcal{E}$ to accumulate knowledge about two tested terms. For the λ-calculus [19], $\mathcal{E}$ records the values (v_0, v_1) the tested terms reduce to, if they exist. We can then compare v_0 and v_1 at any time by passing them arguments built from $\mathcal{E}$. In $\lambda_{\mathcal{S}}$, we have to consider stuck terms as well; therefore, environments may also contain pairs of stuck terms, and we can test those by building pure contexts from $\mathcal{E}$.

Formally, an environment $\mathcal{E}$ is a relation on closed normal forms which relates values with values and stuck terms with stuck terms; e.g., the identity environment $\mathcal{I}$ is $\{(t,t) \mid t \text{ is a normal form}\}$. An environmental relation $\mathcal{X}$ is a set of environments $\mathcal{E}$, and triples $(\mathcal{E}, t_0, t_1)$, where t_0 and t_1 are closed. We write $t_0\ \mathcal{X}_{\mathcal{E}}\ t_1$ as a shorthand for $(\mathcal{E}, t_0, t_1) \in \mathcal{X}$; roughly, it means that we test t_0 and t_1 with the knowledge $\mathcal{E}$. The *open extension* of $\mathcal{X}$, written $\mathcal{X}^\circ$, is defined as follows: if $\overrightarrow{x} = \mathsf{fv}(t_0) \cup \mathsf{fv}(t_1)$[4], then we write $t_0\ \mathcal{X}_{\mathcal{E}}{}^\circ\ t_1$ if $\lambda\overrightarrow{x}.t_0\ \mathcal{X}_{\mathcal{E}}\ \lambda\overrightarrow{x}.t_1$.

[4] Given a metavariable m, we write $\overrightarrow{m}$ for a set of entities denoted by m.

Definition 6. *A relation $\mathcal{X}$ is an environmental bisimulation if*

1. *$t_0\ \mathcal{X}_\mathcal{E}\ t_1$ implies:*
   (a) *if $t_0 \to_\mathrm{v} t_0'$, then $t_1 \to_\mathrm{v}^* t_1'$ and $t_0'\ \mathcal{X}_\mathcal{E}\ t_1'$;*
   (b) *if $t_0 = v_0$, then $t_1 \to_\mathrm{v}^* v_1$ and $\mathcal{E} \cup \{(v_0, v_1)\} \in \mathcal{X}$;*
   (c) *if t_0 is stuck, then $t_1 \to_\mathrm{v}^* t_1'$ with t_1' stuck, and $\mathcal{E} \cup \{(t_0, t_1')\} \in \mathcal{X}$;*
   (d) *the converse of the above conditions on t_1;*
2. *$\mathcal{E} \in \mathcal{X}$ implies:*
   (a) *if $\lambda x.t_0\ \mathcal{E}\ \lambda x.t_1$ and $v_0\ \widehat{\mathcal{E}}\ v_1$, then $t_0\{v_0/x\}\ \mathcal{X}_\mathcal{E}\ t_1\{v_1/x\}$;*
   (b) *if $E_0[\mathcal{S}k.t_0]\ \mathcal{E}\ E_1[\mathcal{S}k.t_1]$ and $E_0'\ \widetilde{\mathcal{E}}\ E_1'$, then $\langle t_0\{\lambda x.\langle E_0'[E_0[x]]\rangle/k\}\rangle\ \mathcal{X}_\mathcal{E}$ $\langle t_1\{\lambda x.\langle E_1'[E_1[x]]\rangle/k\}\rangle$ for a fresh x.*

Environmental bisimilarity, written $\approx$, is the largest environmental bisimulation. To prove that two terms t_0 and t_1 are equivalent, we want to relate them without any predefined knowledge, i.e., we want to prove that $t_0 \approx_\emptyset t_1$ holds; we also write $\simeq$ for $\approx_\emptyset$.

The first part of the definition makes the bisimulation game explicit for t_0, t_1, while the second part focuses on environments $\mathcal{E}$. If t_0 is a normal form, then t_1 has to evaluate to a normal form of the same kind, and we extend the environment with the newly acquired knowledge. We then compare values in $\mathcal{E}$ (clause (2a)) by applying them to arguments built from $\mathcal{E}$, as in the λ-calculus [19]. Similarly, we test stuck terms in $\mathcal{E}$ by putting them within contexts $\langle E_0'\rangle$, $\langle E_1'\rangle$ built from $\mathcal{E}$ (clause (2b)) to trigger the capture. This reminds the way we test values and stuck terms with applicative bisimilarity [4], except that applicative bisimilarity tests both values or stuck terms with the same argument or context. Using different entities (as in Definition 6) makes bisimulation proofs harder, but it simplifies the proof of congruence of the environmental bisimilarity.

Example 1. We have $\langle(\lambda x.t_0)\,\langle t_1\rangle\rangle \simeq (\lambda x.\langle t_0\rangle)\,\langle t_1\rangle$, because the relation $\mathcal{X} = \{(\emptyset, \langle(\lambda x.t)\,\langle t'\rangle\rangle, (\lambda x.\langle t\rangle)\,\langle t'\rangle), (\emptyset, \langle(\lambda x.t)\,v\rangle, (\lambda x.\langle t\rangle)\,v)\} \cup \{(\mathcal{E}, t, t) \mid \mathcal{E} \subseteq \mathcal{I}\} \cup \{\mathcal{E} \mid \mathcal{E} \subseteq \mathcal{I}\}$ is a bisimulation. Indeed, if $\langle t'\rangle$ evaluates to v, then $\langle(\lambda x.t)\,\langle t'\rangle\rangle \to_\mathrm{v}^* \langle(\lambda x.t)\,v\rangle$ and $(\lambda x.\langle t\rangle)\,\langle t'\rangle \to_\mathrm{v}^* (\lambda x.\langle t\rangle)\,v$, which both reduce to $\langle t\{v/x\}\rangle$.

As usual with environmental relations, the candidate relation $\mathcal{X}$ in the above example could be made simpler with the help of up-to techniques.

Definition 6 is written in the small-step style, because each reduction step from t_0 has to be matched by t_1. In the big-step style, we are concerned only with evaluations to normal forms.

Definition 7. *A relation $\mathcal{X}$ is a big-step environmental bisimulation if $t_0\ \mathcal{X}_\mathcal{E}\ t_1$ implies:*

1. *$t_0\ \mathcal{X}_\mathcal{E}\ t_1$ implies:*
   (a) *if $t_0 \to_\mathrm{v}^* v_0$, then $t_1 \to_\mathrm{v}^* v_1$ and $\mathcal{E} \cup \{(v_0, v_1)\} \in \mathcal{X}$;*
   (b) *if $t_0 \to_\mathrm{v}^* t_0'$ with t_0' stuck, then $t_1 \to_\mathrm{v}^* t_1'$, t_1' stuck, and $\mathcal{E} \cup \{(t_0', t_1')\} \in \mathcal{X}$;*
   (c) *the converse of the above conditions on t_1;*
2. *$\mathcal{E} \in \mathcal{X}$ implies:*
   (a) *if $\lambda x.t_0\ \mathcal{E}\ \lambda x.t_1$ and $v_0\ \widehat{\mathcal{E}}\ v_1$, then $t_0\{v_0/x\}\ \mathcal{X}_\mathcal{E}\ t_1\{v_1/x\}$;*

(b) if $E_0[\mathcal{S}k.t_0]\ \mathcal{E}\ E_1[\mathcal{S}k.t_1]$ *and* $E'_0\ \widetilde{\mathcal{E}}\ E'_1$*, then* $\langle t_0\{\lambda x.\langle E'_0[E_0[x]]\rangle/k\}\rangle\ \mathcal{X}_\mathcal{E}$ $\langle t_1\{\lambda x.\langle E'_1[E_1[x]]\rangle/k\}\rangle$ *for a fresh* x*.*

Lemma 4. *If* $\mathcal{X}$ *is a big-step environmental bisimulation, then* $\mathcal{X} \subseteq \approx$*.*

Big-step relations can be more convenient to use when we know the result of the evaluation, as in Example 1, or as in the following one.

Example 2. We have $\langle\langle t\rangle\rangle \simeq \langle t\rangle$. Indeed, we can show that $\langle\langle t\rangle\rangle \to_{\mathrm{v}}^* v$ iff $\langle t\rangle \to_{\mathrm{v}}^* v$, therefore $\{(\emptyset, \langle\langle t\rangle\rangle, \langle t\rangle)\} \cup \{(\mathcal{E}, t, t) \mid \mathcal{E} \subseteq \mathcal{I}\} \cup \{\mathcal{E} \mid \mathcal{E} \subseteq \mathcal{I}\}$ is a big-step environmental bisimulation.

We use the following results in the rest of the paper.

Lemma 5 (Weakening). *If* $t_0 \approx_\mathcal{E} t_1$ *and* $\mathcal{E}' \subseteq \mathcal{E}$ *then* $t_0 \approx_{\mathcal{E}'} t_1$*.*

A smaller environment is a weaker constraint, because we can build less arguments and contexts to test the normal forms in $\mathcal{E}$. The proof is as in [19]. Lemma 6 states that reduction (and therefore, evaluation) is included in $\simeq$.

Lemma 6. *If* $t_0 \to_{\mathrm{v}} t'_0$*, then* $t_0 \simeq t'_0$*.*

3.3 Soundness and Completeness

We now prove soundness and completeness of $\simeq$ w.r.t. contextual equivalence. Because the proofs follow the same steps as for the λ-calculus [19], we only give here the main lemmas and sketch their proofs. The complete proofs can be found in [6]. First, we need some basic up-to techniques, namely up-to environment (which allows bigger environments in the bisimulation clauses) and up-to bisimilarity (which allows for limited uses of $\simeq$ in the bisimulation clauses), whose definitions and proofs of soundness are classic [19].

With these tools, we can prove that $\simeq$ is sound and complete w.r.t. contextual equivalence. For a relation $\mathcal{R}$ on terms, we write $\mathcal{R}^{\,\mathrm{nf}}$ for its restriction to closed normal forms. The first step consists in proving congruence for normal forms, and also for any terms but only w.r.t. evaluation contexts.

Lemma 7. *Let* t_0*,* t_1 *be normal forms. If* $t_0 \approx_\mathcal{E} t_1$*, then* $C[t_0] \approx_\mathcal{E} C[t_1]$*.*

Lemma 8. *If* $t_0 \approx_\mathcal{E} t_1$*, then* $F[t_0] \approx_\mathcal{E} F[t_1]$*.*

Lemmas 7 and 8 are proved simultaneously by showing that, for any environmental bisimulation $\mathcal{Y}$, the relation

$$\mathcal{X} = \{(\widehat{\mathcal{E}}^{\,\mathrm{nf}}, F_0[t_0], F_1[t_1]) \mid t_0\ \mathcal{Y}_\mathcal{E}\ t_1, F_0\ \widetilde{\mathcal{E}}\ F_1\} \cup \{(\widehat{\mathcal{E}}^{\,\mathrm{nf}}, t_0, t_1) \mid \mathcal{E} \in \mathcal{Y}, t_0\ \widehat{\mathcal{E}}\ t_1\} \cup \{\widehat{\mathcal{E}}^{\,\mathrm{nf}} \mid \mathcal{E} \in \mathcal{Y}\}$$

is a bisimulation up-to environment. Informally, the elements of the first set of $\mathcal{X}$ reduce to elements of the second set of $\mathcal{X}$, and we then prove the bisimulation property for these elements by induction on $t_0\ \widehat{\mathcal{E}}\ t_1$. We can then prove the main congruence lemma.

Lemma 9. *$t_0 \simeq t_1$ implies $C[t_0] \approx_{\widehat{\simeq}^{\mathrm{nf}}} C[t_1]$.*

We show that $\{(\widehat{\simeq}^{\mathrm{nf}}, t_0, t_1) \mid t_0 \,\widehat{\simeq}\, t_1\} \cup \{\widehat{\simeq}^{\mathrm{nf}}\}$ is a bisimulation up-to bisimilarity by induction on $t_0 \,\widehat{\simeq}\, t_1$. By weakening (Lemma 5), we can deduce from Lemma 9 that $\simeq$ is a congruence, and therefore is sound w.r.t. $\approx_c$.

Corollary 1 (Soundness). *We have $\simeq \subseteq \approx_c$.*

The relation $\simeq$ is also complete w.r.t. contextual equivalence.

Theorem 1 (Completeness). *We have $\approx_c \subseteq \simeq$.*

The proof is by showing that $\{(\approx_c{}^{\mathrm{nf}}, t_0, t_1) \mid t_0 \approx_c t_1\} \cup \{\approx_c{}^{\mathrm{nf}}\}$ is a big-step bisimulation, using Lemma 3 as an alternate definition for $\approx_c$.

3.4 Bisimulation up to context

Equivalence proofs based on environmental bisimilarity can be simplified by using up-to techniques, such as up to reduction, up to expansion, and up to context [19]. We only discuss the last, since the first two can be defined and proved sound in $\lambda_{\mathcal{S}}$ without issues. Bisimulations up to context may factor out a common context from the tested terms. Formally, we define the context closure of $\mathcal{X}$, written $\overline{\mathcal{X}}$, as follows: we have $t_0 \,\overline{\mathcal{X}_{\mathcal{E}}}\, t_1$ if

- either $t_0 = F_0[t_0']$, $t_1 = F_1[t_1']$, $t_0' \,\mathcal{X}_{\mathcal{E}}\, t_1'$, and $F_0 \,\widetilde{\mathcal{E}}\, F_1$;
- or $t_0 \,\widehat{\mathcal{E}}\, t_1$.

Note that terms t_0' and t_1' (related by $\mathcal{X}_{\mathcal{E}}$) can be put into evaluation contexts only, while normal forms (related by $\mathcal{E}$) can be put in any contexts. This restriction to evaluation contexts in the first case is usual in the definition of up-to context techniques for environmental relations [19, 22, 20, 18].

Definition 8. *A relation $\mathcal{X}$ is an environmental bisimulation up to context if*

1. *$t_0 \,\mathcal{X}_{\mathcal{E}}\, t_1$ implies:*
   (a) *if $t_0 \to_{\mathrm{v}} t_0'$, then $t_1 \to_{\mathrm{v}}^* t_1'$ and $t_0' \,\overline{\mathcal{X}_{\mathcal{E}}}\, t_1'$;*
   (b) *if $t_0 = v_0$, then $t_1 \to_{\mathrm{v}}^* v_1$ and $\mathcal{E} \cup \{(v_0, v_1)\} \subseteq \widehat{\mathcal{E}'}^{\mathrm{nf}}$ for some $\mathcal{E}' \in \mathcal{X}$;*
   (c) *if t_0 is stuck, then $t_1 \to_{\mathrm{v}}^* t_1'$ with t_1' stuck, and $\mathcal{E} \cup \{(t_0, t_1')\} \subseteq \widehat{\mathcal{E}'}^{\mathrm{nf}}$ for some $\mathcal{E}' \in \mathcal{X}$;*
   (d) *the converse of the above conditions on t_1;*
2. *$\mathcal{E} \in \mathcal{X}$ implies:*
   (a) *if $\lambda x.t_0 \;\mathcal{E}\; \lambda x.t_1$ and $v_0 \,\widehat{\mathcal{E}}\, v_1$, then $t_0\{v_0/x\} \,\overline{\mathcal{X}_{\mathcal{E}}}\, t_1\{v_1/x\}$;*
   (b) *if $E_0[\mathcal{S}k.t_0] \;\mathcal{E}\; E_1[\mathcal{S}k.t_1]$ and $E_0' \,\widetilde{\mathcal{E}}\, E_1'$, then $\langle t_0\{\lambda x.\langle E_0'[E_0[x]]\rangle/k\}\rangle \;\overline{\mathcal{X}_{\mathcal{E}}}\; \langle t_1\{\lambda x.\langle E_1'[E_1[x]]\rangle/k\}\rangle$ for a fresh x.*

Lemma 10. *If $\mathcal{X}$ is an environmental bisimulation up to context, then $\mathcal{X} \subseteq \approx$.*

The soundness proof is the same as in [19]. While this definition is enough to simplify proofs in the λ-calculus case, it is not that helpful in $\lambda_{\mathcal{S}}$, because of the restriction to evaluation contexts (first item of the definition of $\overline{\mathcal{X}}$). In the λ-calculus, when a term t reduces within an evaluation context, the context is not affected, hence Definition 8 is enough to help proving interesting equivalences. It is not the case in $\lambda_{\mathcal{S}}$, as (a part of) the evaluation context can be captured.

Indeed, suppose we want to construct a candidate relation $\mathcal{X}$ to prove the β_Ω axiom, i.e., $E[t]$ is equivalent to $(\lambda x.E[x])\ t$, assuming $x \notin \mathsf{fv}(E)$. The problematic case is when t is a stuck term $E_0[\mathcal{S}k.t_0]$; we have to add the stuck terms $(\lambda x.E[x])\ E_0[\mathcal{S}k.t_0]$ and $E[E_0[\mathcal{S}k.t_0]]$ to an environment $\mathcal{E}$ of $\mathcal{X}$. For $\mathcal{X}$ to be a bisimulation, we then have to prove that for all $E_1\ \widetilde{\mathcal{E}}\ E_2$, we have $\langle t_0\{\lambda y.\langle E_1[(\lambda x.E[x])\ E_0[y]]\rangle/k\}\rangle\ \mathcal{X}_{\mathcal{E}}\ \langle t_0\{\lambda y.\langle E_2[E[E_0[y]]]\rangle/k\}\rangle$. At this point, we would like to use the up-to context technique, because the subterms $(\lambda x.E[x])\,E_0[y]$ and $E[E_0[y]]$ are similar to the terms we want to relate (they can be written $(\lambda x.E[x])\ t''$ and $E[t'']$ with $t'' = E_0[y]$). However, we have at best $\langle t_0\{\lambda y.\langle E_1[(\lambda x.E[x])\ E_0[y]]\rangle/k\}\rangle\ \widehat{\mathcal{X}_{\mathcal{E}}^{\circ}}\ \langle t_0\{\lambda y.\langle E_2[E[E_0[y]]]\rangle/k\}\rangle$ (and not $\overline{\mathcal{X}_{\mathcal{E}}}$), because (i) $(\lambda x.E[x])\ E_0[y]$ and $E[E_0[y]]$ are open terms, and (ii) t_0 can be any term, so $(\lambda x.E[x])\,E_0[y]$ and $E[E_0[y]]$ can be put in any context, not necessarily in an evaluation one. Therefore, Definition 8 cannot help there.

Problem (ii) could be somewhat dealt with in the particular case of the β_Ω axiom by changing clause (2b) of Definition 8 into

(b) if $E_0[\mathcal{S}k.t_0]\ \mathcal{E}\ E_1[\mathcal{S}k.t_1]$ and $E'_0\ \widetilde{\mathcal{X}_{\mathcal{E}}}\ E'_1$, then $\langle t_0\{\lambda x.\langle E'_0[E_0[x]]\rangle/k\}\rangle\ \widehat{\mathcal{X}_{\mathcal{E}}}\ \langle t_1\{\lambda x.\langle E'_1[E_1[x]]\rangle/k\}\rangle$ for a fresh x.

and similarly for clause (2a). In plain text, we build the testing contexts E'_0, E'_1 from $\mathcal{X}_{\mathcal{E}}$ (instead of $\mathcal{E}$), and the resulting terms have to be in $\widehat{\mathcal{X}_{\mathcal{E}}}$ (without any evaluation context restriction). The resulting notion of bisimulation up to context is sound. The new clause would be more difficult to establish in general than the original one (of Definition 8), because it tests more pairs of contexts. However, for the β_Ω axiom, we would have to prove that for all $E_1\ \widehat{\mathcal{X}_{\mathcal{E}}}\ E_2$, $\langle t_0\{\lambda y.\langle E_1[(\lambda x.E[x])\ E_0[y]]\rangle/k\}\rangle\ \widehat{\mathcal{X}_{\mathcal{E}}}\ \langle t_0\{\lambda y.\langle E_2[E[E_0[y]]]\rangle/k\}\rangle$ holds; it would be easy, except $(\lambda x.E[x])\ E_0[y]$ and $E[E_0[y]]$ are open terms (problem (i)).

Problem (i) seems harder to fix, because for $(\lambda x.E[x])\ E_0[y]\ \mathcal{X}_{\mathcal{E}}{}^{\circ}\ E[E_0[y]]$ to hold, we must have $(\lambda x.E[x])\ E_0[v_0]\ \mathcal{X}_{\mathcal{E}}\ E[E_0[v_1]]$ for all $v_0\ \widehat{\mathcal{E}}\ v_1$. Because E_0 can be anything, it means that we must have $(\lambda x.E[x])\,t'_0\ \mathcal{X}_{\mathcal{E}}\ E[t'_1]$ with $t'_0\ \widehat{\mathcal{E}}\ t'_1$; t'_0 and t'_1 are plugged in different contexts, therefore bisimulation up to context (which factors out only a common context) cannot help us there; a new kind of up-to technique is required.

The β_Ω axiom example suggests that we need more powerful up-to techniques for environmental bisimilarity for delimited control; we leave these potential improvements as a future work. Note that we do not have such issues with up-to techniques for normal form bisimilarity: it relates open terms without having to replace their free variables, and normal form bisimulation up to context is not restricted to evaluation contexts only. But even if environmental bisimulation

up to context is not as helpful as wished, it still simplifies equivalence proofs, as we can see with the next example.

Example 3. In [7], a variant of Turing's call-by-value fixed point combinators using shift and reset has been proposed. Let $\theta = \lambda xy.y\ (\lambda z.x\ x\ y\ z)$. We prove that $t_0 = \theta\ \theta$ is bisimilar to its variant $t_1 = \langle \theta\ \mathcal{S}k.k\ k \rangle$. Let $\theta' = \lambda x.\langle \theta\ x \rangle$, $v_0 = \lambda y.y\ (\lambda z.\theta\ \theta\ y\ z)$, and $v_1 = \lambda y.y\ (\lambda z.\theta'\ \theta'\ y\ z)$. We define $\mathcal{E}$ inductively such that $v_0\ \mathcal{E}\ v_1$, and if $v_0'\ \widehat{\mathcal{E}}\ v_1'$, then $\lambda z.\theta\ \theta\ v_0'\ z\ \mathcal{E}\ \lambda z.\theta'\ \theta'\ v_1'\ z$. Then $\mathcal{X} = \{(\mathcal{E}, t_0, t_1), (\mathcal{E}, t_0, \theta'\ \theta'), \mathcal{E}\}$ is a (big-step) bisimulation up to context. Indeed, we have $t_0 \Downarrow_{\mathrm{v}} v_0$, $t_1 \Downarrow_{\mathrm{v}} v_1$, and $\theta'\ \theta' \Downarrow_{\mathrm{v}} v_1$, therefore clause (1b) of Definition 8 is checked for both pairs. We now check clause (2a), first for $v_0\ \mathcal{E}\ v_1$. For all $v_0'\ \widehat{\mathcal{E}}\ v_1'$, we have $v_0'\ (\lambda z.\theta\ \theta\ v_0'\ z)\ \widehat{\mathcal{E}}\ v_1'\ (\lambda z.\theta'\ \theta'\ v_1'\ z)$ (because $\lambda z.\theta\ \theta\ v_0'\ z\ \mathcal{E}\ \lambda z.\theta'\ \theta'\ v_1'\ z$), hence the result holds. Next, let $\lambda z.\theta\ \theta\ v_0'\ z\ \mathcal{E}\ \lambda z.\theta'\ \theta'\ v_1'\ z$ (with $v_0'\ \widehat{\mathcal{E}}\ v_1'$), and let $v_0''\ \widehat{\mathcal{E}}\ v_1''$. We have to check that $\theta\ \theta\ v_0'\ v_0''\ \overline{\mathcal{X}_{\mathcal{E}}}\ \theta'\ \theta'\ v_1'\ v_1''$, which is true, because $\theta\ \theta\ \mathcal{X}_{\mathcal{E}}\ \theta'\ \theta'$, and $\Box\ v_0'\ v_0''\ \widetilde{\mathcal{E}}\ \Box\ v_1'\ v_1''$.

4 Environmental Relations for the Original Semantics

The original CPS semantics for shift and reset [8] as well as the corresponding reduction semantics [3] assume that terms can be considered as programs to be executed, only when surrounded by a top-level reset. In this section, we present a CPS-compatible bisimulation theory that takes such a requirement into account. Henceforth, we call *programs*, ranged over by p, terms of the form $\langle t \rangle$.

4.1 Contextual Equivalence

To reflect the fact that terms are executed within an enclosing reset, the contextual equivalence we consider in this section tests terms in contexts of the form $\langle C \rangle$ only. Because programs cannot reduce to stuck terms, the only possible observable action is evaluation to values. We therefore define contextual equivalence for programs as follows.

Definition 9. *Let t_0, t_1 be terms. We write $t_0 \dot{\approx}_c t_1$ if for all C such that $\langle C[t_0] \rangle$ and $\langle C[t_1] \rangle$ are closed, $\langle C[t_0] \rangle \Downarrow_{\mathrm{v}} v_0$ implies $\langle C[t_1] \rangle \Downarrow_{\mathrm{v}} v_1$, and conversely for $\langle C[t_1] \rangle$.*

Note that $\dot{\approx}_c$ is defined on all terms, not just programs. It is easy to check that $\approx_c$ is more discriminative than $\dot{\approx}_c$. We will see in Section 4.4 that this inclusion is in fact strict.

Lemma 11. *We have $\approx_c \subseteq \dot{\approx}_c$.*

4.2 Definition and Properties

We now propose a definition of environmental bisimulation adapted to programs (but defined on all terms, like $\dot{\approx}_c$). Because stuck terms are no longer observed, environments $\mathcal{E}$ henceforth relate only values. Similarly, we write $\mathcal{R}^{\mathrm{v}}$ for the restriction of a relation $\mathcal{R}$ on terms to pairs of closed values.

Definition 10. *A relation $\mathcal{X}$ is an environmental bisimulation for programs if*

1. *if $t_0\ \mathcal{X}_{\mathcal{E}}\ t_1$ and t_0 and t_1 are not both programs, then for all $E_0\ \widetilde{\mathcal{E}}\ E_1$, we have $\langle E_0[t_0]\rangle\ \mathcal{X}_{\mathcal{E}}\ \langle E_1[t_1]\rangle$;*
2. *if $p_0\ \mathcal{X}_{\mathcal{E}}\ p_1$*
   (a) if $p_0 \to_{\mathrm{v}} p_0'$, then $p_1 \to_{\mathrm{v}}^* p_1'$ and $p_0'\ \mathcal{X}_{\mathcal{E}}\ p_1'$;
   (b) if $p_0 \to_{\mathrm{v}} v_0$, then $p_1 \to_{\mathrm{v}}^* v_1$, and $\{(v_0, v_1)\} \cup \mathcal{E} \in \mathcal{X}$;
   (c) the converse of the above conditions on p_1;
3. *for all $\mathcal{E} \in \mathcal{X}$, if $\lambda x.t_0\ \mathcal{E}\ \lambda x.t_1$ and $v_0\ \widehat{\mathcal{E}}\ v_1$, then $t_0\{v_0/x\}\ \mathcal{X}_{\mathcal{E}}\ t_1\{v_1/x\}$.*

Environmental bisimilarity for programs, written $\dot{\approx}$, is the largest environmental bisimulation for programs. As before, the relation $\dot{\approx}_{\emptyset}$, also written $\dot{\simeq}$, is candidate to characterize $\dot{\approx}_c$.

Clauses (2) and (3) of Definition 10 deal with programs and environment in a classical way (as in plain λ-calculus). The problematic case is when relating terms t_0 and t_1 that are not both programs (clause (1)). Indeed, one of them may be stuck, and therefore we have to test them within some contexts $\langle E_0\rangle$, $\langle E_1\rangle$ (built from $\mathcal{E}$) to potentially trigger a capture that otherwise would not happen. We cannot require both terms to be stuck, as in clause (2b) of Definition 6, because a stuck term can be equivalent to a term free from control effect. E.g., we will see that $v \dot{\simeq} \mathcal{S}k.k\ v$, provided that $k \notin \mathsf{fv}(v)$.

Example 4. Suppose we want to prove $\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle \dot{\simeq} (\lambda x.\langle t_0\rangle)\ \langle t_1\rangle$ (as in Example 1). Because $(\lambda x.\langle t_0\rangle)\,\langle t_1\rangle$ is not a program, we have to put both terms into a context first: we have to change the candidate relation of Example 1 into $\mathcal{X} = \{(\emptyset, \langle(\lambda x.t_0)\ \langle t_1\rangle\rangle, (\lambda x.\langle t_0\rangle)\,\langle t_1\rangle)\} \cup \{(\emptyset, \langle E[\langle(\lambda x.t_0)\ \langle t_1\rangle\rangle]\rangle, \langle E[(\lambda x.\langle t_0\rangle)\ \langle t_1\rangle]\rangle)\} \cup \{(\emptyset, \langle E[\langle(\lambda x.t_0)\ v\rangle]\rangle, \langle E[(\lambda x.\langle t_0\rangle)\ v]\rangle)\} \cup \{(\mathcal{E}, t, t)\ \mid\ \mathcal{E} \subseteq \mathcal{I}\} \cup \{\mathcal{E}\ \mid\ \mathcal{E} \subseteq \mathcal{I}\}$. In contrast, to prove $\langle\langle t\rangle\rangle \dot{\simeq} \langle t\rangle$, we do not have to change the candidate relation of Example 2, since both terms are programs.

We can give a definition of big-step bisimulation by removing clause (2a) and changing $\to_{\mathrm{v}}$ into $\to_{\mathrm{v}}^*$ in clause (2b). Lemmas 5 and 6 can also be extended to $\dot{\approx}$ and $\dot{\simeq}$. The next lemma shows that $\simeq$ is more discriminative than $\dot{\simeq}$.

Lemma 12. *We have $\simeq\ \subseteq\ \dot{\simeq}$.*

A consequence of Lemma 12 is that we can use Definition 6 as a proof technique for $\dot{\simeq}$. E.g., we have directly $\langle(\lambda x.t_0)\,\langle t_1\rangle\rangle \dot{\simeq} (\lambda x.\langle t_0\rangle)\,\langle t_1\rangle$, because $\langle(\lambda x.t_0)\,\langle t_1\rangle\rangle \simeq (\lambda x.\langle t_0\rangle)\,\langle t_1\rangle$.

4.3 Soundness and Completeness

We sketch the proofs of soundness and completeness of $\dot{\simeq}$ w.r.t. $\dot{\approx}_c$; see [6] for the complete proofs. The soundness proof follows the same scheme as in Section 3.3, with some necessary adjustments. As before, we need up-to environment and up-to bisimilarity techniques to prove the following lemmas.

Lemma 13. *If $v_0 \dot{\approx}_{\mathcal{E}} v_1$, then $C[v_0] \dot{\approx}_{\mathcal{E}} C[v_1]$.*

Lemma 14. *If $t_0 \mathrel{\dot{\approx}}_{\mathcal{E}} t_1$, then $F[t_0] \mathrel{\dot{\approx}}_{\mathcal{E}} F[t_1]$.*

We prove Lemmas 13 and 14 by showing that a relation similar to the relation $\mathcal{X}$ defined in Section 3.3 is a bisimulation up to environment. We then want to prove the main congruence lemma, akin to Lemma 9, by showing that $\mathcal{Y} = \{(\mathrel{\dot{\widehat{\simeq}}}^{\mathrm{v}}, t_0, t_1) \mid t_0 \mathrel{\dot{\widehat{\simeq}}} t_1\} \cup \{\mathrel{\dot{\widehat{\simeq}}}^{\mathrm{v}}\}$ is a bisimulation up to bisimilarity. However, we can no longer proceed by induction on $t_0 \mathrel{\dot{\widehat{\simeq}}} t_1$, as for Lemma 9. Indeed, if $p_0 = \langle t_0 \rangle$, $p_1 = \langle t_1 \rangle$ with $t_0 \mathrel{\dot{\widehat{\simeq}}} t_1$, and if t_0 is a stuck term, then p_0 reduces to some term, but the induction hypothesis does not tell us anything about t_1. To circumvent this, we decompose related programs into related subcomponents.

Lemma 15. *If $p_0 \mathrel{\dot{\widehat{\simeq}}} p_1$, then either $p_0 \mathrel{\dot{\simeq}} p_1$, or one of the following holds:*

- *$p_0 = \langle v_0 \rangle$;*
- *$p_0 = F_0[\langle E_0[t_0] \rangle]$, $p_1 = F_1[\langle E_1[t_1] \rangle]$, $F_0 \mathrel{\dot{\widetilde{\simeq}}} F_1$, $E_0 \mathrel{\dot{\widetilde{\simeq}}} E_1$, $t_0 \mathrel{\dot{\simeq}} t_1$ and $t_0 \to_{\mathrm{v}} t_0'$ or t_0 is stuck;*
- *$p_0 = F_0[\langle E_0[r_0] \rangle]$, $p_1 = F_1[\langle E_1[t_1] \rangle]$, $F_0 \mathrel{\dot{\widetilde{\simeq}}} F_1$, $E_0 \mathrel{\dot{\widetilde{\simeq}}} E_1$, $r_0 \mathrel{\dot{\widehat{\simeq}}} t_1$ but $r_0 \not\mathrel{\ddot{\simeq}} t_1$.*

Lemma 15 generalizes Lemma 2 to related programs: we know p_0 can be decomposed into contexts F, $\langle E \rangle$, and a redex r, and we relate these subterms to p_1. We can then prove that $\mathcal{Y}$ (defined above) is a bisimulation up to bisimilarity, by showing that, in each case described by Lemma 15, p_0 and p_1 reduce to terms related by $\mathcal{Y}$. From this, we deduce $\mathrel{\dot{\simeq}}$ is a congruence, and is sound w.r.t. $\mathrel{\dot{\approx}}_c$.

Lemma 16. *$t_0 \mathrel{\dot{\simeq}} t_1$ implies $C[t_0] \mathrel{\dot{\approx}}_{\dot{\widehat{\simeq}}^{\mathrm{v}}} C[t_1]$.*

Corollary 2 (Soundness). *We have $\mathrel{\dot{\simeq}} \subseteq \mathrel{\dot{\approx}}_c$.*

Remark 1. Following the ideas behind Definition 10, one can define an applicative bisimilarity $\mathcal{B}$ for programs. However, proving that $\mathcal{B}$ is sound seems more complex than for $\mathrel{\dot{\simeq}}$. We remind that the soundness proof of an applicative bisimilarity consists in showing that a relation called the *Howe's closure* $\mathcal{B}^\bullet$ is an applicative bisimulation. To this end, we need a version of Lemma 15 for $\mathcal{B}^\bullet$. However, $\mathcal{B}^\bullet$ is inductively defined as the smallest congruence which contains $\mathcal{B}$ and satisfies $\mathcal{B}^\bullet \mathcal{B} \subseteq \mathcal{B}^\bullet$ (1), and condition (1) makes it difficult to write a decomposition lemma for $\mathcal{B}^\bullet$ similar to Lemma 15.

We prove completeness of $\mathrel{\dot{\simeq}}$ by showing that the relation $\mathrel{\ddot{\approx}}_c$, defined below, coincides with $\mathrel{\dot{\approx}}_c$ and $\mathrel{\dot{\simeq}}$. By doing so, we also prove a context lemma for $\mathrel{\dot{\approx}}_c$.

Definition 11. *Let t_0, t_1 be closed terms. We write $t_0 \mathrel{\ddot{\approx}}_c t_1$ if for all closed F, $\langle F[t_0] \rangle \Downarrow_{\mathrm{v}} v_0$ implies $\langle F[t_1] \rangle \Downarrow_{\mathrm{v}} v_1$, and conversely for $\langle F[t_1] \rangle$.*

By definition, we have $\mathrel{\dot{\approx}}_c \subseteq \mathrel{\ddot{\approx}}_c$. With the same proof technique as in Section 3.3, we prove the following lemma.

Lemma 17 (Completeness). *We have $\mathrel{\ddot{\approx}}_c \subseteq \mathrel{\dot{\simeq}}$.*

With Lemma 17 and Corollary 2, we have $\mathrel{\dot{\approx}}_c \subseteq \mathrel{\ddot{\approx}}_c \subseteq \mathrel{\dot{\simeq}} \subseteq \mathrel{\dot{\approx}}_c$. Defining up-to context for programs is possible, with the same limitations as in Section 3.4.

4.4 Examples

We illustrate the differences between $\simeq$ and $\dot{\simeq}$, by giving some examples of terms related by $\dot{\simeq}$, but not by $\simeq$. First, note that $\dot{\simeq}$ relates non-terminating terms with stuck non-terminating terms.

Lemma 18. *We have* $\Omega \dot{\simeq} \mathcal{S}k.\Omega$.

The relation $\{(\emptyset, \Omega, \mathcal{S}k.\Omega), (\emptyset, \langle E[\Omega]\rangle, \langle E[\mathcal{S}k.\Omega]\rangle), (\emptyset, \langle E[\Omega]\rangle, \langle\Omega\rangle)\}$ is a bisimulation for programs. Lemma 18 does not hold with $\simeq$ because Ω is not stuck.

As wished, $\dot{\simeq}$ satisfies the only axiom of [14] not satisfied by $\simeq$.

Lemma 19. *If* $k \notin \mathsf{fv}(t)$*, then* $t \dot{\simeq}^{\circ} \mathcal{S}k.k\, t$.

We sketch the proof for t closed; for the general case, see [6]. We prove that $\{(\emptyset, t, \mathcal{S}k.k\, t), (\emptyset, \langle E[t]\rangle, \langle E[\mathcal{S}k.k\, t]\rangle)\} \cup \simeq$ is a bisimulation for programs. Indeed, we have $\langle E[\mathcal{S}k.k\, t]\rangle \to_{\mathrm{v}} \langle(\lambda x.\langle E[x]\rangle)\, t\rangle$, and because $\simeq$ verifies the β_Ω axiom ($\simeq$ is complete, and $\approx_c$ verifies the β_Ω axiom [4]), we know that $\langle(\lambda x.\langle E[x]\rangle)\, t\rangle \simeq \langle\langle E[t]\rangle\rangle$ holds. From Example 2, we have $\langle\langle E[t]\rangle\rangle \simeq \langle E[t]\rangle$, therefore we have $\langle E[\mathcal{S}k.k\, t]\rangle \simeq \langle E[t]\rangle$.

Consequently, $\dot{\simeq}^{\circ}$ is complete w.r.t. $\equiv$.

Corollary 3. *We have* $\equiv \subseteq \dot{\simeq}^{\circ}$.

As a result, we can use $\equiv$ (restricted to closed terms) as a proof technique for $\dot{\simeq}$. E.g., the following equivalence can be derived from the axioms [14].

Lemma 20. *If* $k \notin \mathsf{fv}(t_1)$*, then* $(\lambda x.\mathcal{S}k.t_0)\, t_1 \dot{\simeq} \mathcal{S}k.((\lambda x.t_0)\, t_1)$.

This equivalence does not hold with $\simeq$, because the term on the right is stuck, but the term on the left may not evaluate to a stuck term (if t_1 does not terminate). We can generalize this result as follows, again by using $\equiv$.

Lemma 21. *If* $k \notin \mathsf{fv}(t_1)$ *and* $x \notin \mathsf{fv}(E)$*, then we have* $(\lambda x.E[\mathcal{S}k.t_0])\, t_1 \dot{\simeq} E[\mathcal{S}k.((\lambda x.t_0)\, t_1)]$.

Proving Lemma 19 without the β_Ω axiom and Lemmas 20 and 21 without $\equiv$ requires complex candidate relations (see the proof of Lemma 20 in [6]), because of the lack of powerful enough up-to techniques.

5 Conclusion

We propose sound and complete environmental bisimilarities for two variants of the semantics of $\lambda_{\mathcal{S}}$. For the semantics of Section 3, we now have several bisimilarities, each with its own merit. Normal form bisimilarity [5] (and its up-to techniques) leads to minimal proof obligations, however it is not complete, and distinguishes very simple equivalent terms (see Proposition 1 in [5]). Applicative bisimilarity [4] is complete but sometimes requires complex bisimulation proofs (e.g., for the β_Ω axiom). Environmental bisimilarity $\simeq$ (Definition 6) is also complete, can be difficult to use, but this difficulty can be mitigated with up-to

techniques. However, bisimulation up to context is not as helpful as we could hope (see Section 3.4), because we have to manipulate open terms (problem (i)), and the context closure of an environmental relation is restricted to evaluation contexts (problem (ii)). As a result, proving the β_Ω axiom is more difficult with environmental than with applicative bisimilarity. We believe dealing with problem (i) requires new up-to techniques to be developed, and lifting the evaluation context restriction (problem (ii)) would benefit not only for $\lambda_{\mathcal{S}}$, but also for process calculi with passivation [18]; we leave this as a future work.

In contrast, we do not have as many options when considering the semantics of Section 4 (where terms are evaluated within a top-level reset). The environmental bisimilarity of this paper $\dot{\simeq}$ (Definition 10) is the first to be sound and complete w.r.t. Definition 9. As argued in [5] (Section 3.2), normal form bisimilarity cannot be defined on programs without introducing extra quantifications (which defeats the purpose of normal form bisimilarity). Applicative bisimilarity could be defined for programs, but proving its soundness would require a new technique, since the usual one (Howe's method) does not seem to apply (see Remark 1). This confirms that environmental bisimilarity is more flexible than applicative bisimilarity [15]. However, we would like to simplify the quantification over contexts in clause (1) of Definition 10, so we look for sub-classes of terms where this quantification is not mandatory.

Other future works include the study of the behavioral theory of other delimited control operators, like the dynamic ones (e.g., *control* and *prompt* [10] or $shift_0$ and $reset_0$ [7]), but also of abortive control operators, such as *callcc*, for which no sound and complete bisimilarity has been defined so far.

Acknowledgments We thank Małgorzata Biernacka and the anonymous referees for many helpful comments on the presentation of this work.

References

1. K. Asai and Y. Kameyama. Polymorphic delimited continuations. In Z. Shao, editor, *APLAS'07*, number 4807 in LNCS, pages 239–254, Singapore, Dec. 2007. Springer-Verlag.
2. V. Balat, R. D. Cosmo, and M. P. Fiore. Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums. In X. Leroy, editor, *POPL'04*, SIGPLAN Notices, Vol. 39, No. 1, pages 64–76, Venice, Italy, Jan. 2004. ACM Press.
3. M. Biernacka, D. Biernacki, and O. Danvy. An operational foundation for delimited continuations in the CPS hierarchy. *Logical Methods in Computer Science*, 1(2:5):1–39, Nov. 2005.
4. D. Biernacki and S. Lenglet. Applicative bisimulations for delimited-control operators. In L. Birkedal, editor, *FOSSACS'12*, number 7213 in LNCS, pages 119–134, Tallinn, Estonia, Mar. 2012. Springer-Verlag.
5. D. Biernacki and S. Lenglet. Normal form bisimulations for delimited-control operators. In T. Schrijvers and P. Thiemann, editors, *FLOPS'12*, number 7294 in LNCS, pages 47–61, Kobe, Japan, May 2012. Springer-Verlag.

6. D. Biernacki and S. Lenglet. Environmental bisimulations for delimited-control operators, Sept. 2013. Available at `http://hal.inria.fr/hal-00862189`.
7. O. Danvy and A. Filinski. A functional abstraction of typed contexts. DIKU Rapport 89/12, DIKU, Computer Science Department, University of Copenhagen, Copenhagen, Denmark, July 1989.
8. O. Danvy and A. Filinski. Abstracting control. In Wand [24], pages 151–160.
9. R. K. Dybvig, S. Peyton-Jones, and A. Sabry. A monadic framework for delimited continuations. *Journal of Functional Programming*, 17(6):687–730, 2007.
10. M. Felleisen. The theory and practice of first-class prompts. In J. Ferrante and P. Mager, editors, *POPL'88*, pages 180–190, San Diego, California, Jan. 1988. ACM Press.
11. A. Filinski. Representing monads. In H.-J. Boehm, editor, *POPL'94*, pages 446–457, Portland, Oregon, Jan. 1994. ACM Press.
12. R. Hieb, R. K. Dybvig, and C. W. Anderson, III. Subcontinuations. *Lisp and Symbolic Computation*, 5(4):295–326, Dec. 1993.
13. Y. Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007.
14. Y. Kameyama and M. Hasegawa. A sound and complete axiomatization of delimited continuations. In O. Shivers, editor, *ICFP'03*, SIGPLAN Notices, Vol. 38, No. 9, pages 177–188, Uppsala, Sweden, Aug. 2003. ACM Press.
15. V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. *Electronic Notes in Theoretical Computer Science*, 276:215–235, 2011.
16. R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.
17. J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.
18. A. Piérard and E. Sumii. A higher-order distributed calculus with name creation. In *LICS'12*, pages 531–540, Dubrovnik, Croatia, June 2012. IEEE Computer Society Press.
19. D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Transactions on Programming Languages and Systems*, 33(1):1–69, Jan. 2011.
20. N. Sato and E. Sumii. The higher-order, call-by-value applied Pi-calculus. In Z. Hu, editor, *APLAS'09*, volume 5904 of *LNCS*, pages 311–326, Seoul, Korea, Dec. 2009. Springer-Verlag.
21. D. Sitaram and M. Felleisen. Reasoning with continuations II: Full abstraction for models of control. In Wand [24], pages 161–175.
22. E. Sumii. A bisimulation-like proof method for contextual properties in untyped lambda-calculus with references and deallocation. *Theoretical Computer Science*, 411(51-52):4358–4378, 2010.
23. E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1-3):169–192, 2007.
24. M. Wand, editor. *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, Nice, France, June 1990. ACM Press.